

D. REMARKS

In the office action, claim 11 was rejected under 35 U.S.C. 112, second paragraph, as failing to particularly point out and distinctly claim the subject matter. Further, the antecedent basis was not clear. Furthermore, claims 16 and 20-23 were rejected under 35 U.S.C. 112 as failing to invoke 35 U.S.C. 112 6th paragraph by using "means-plus-function" language. Moreover, claims 1-2 were rejected under 35 U.S.C. 102(e), as anticipated by the patent to Ignatius et al U.S patent number 7,209,972 (hereinafter referred to as, "Ignatius"). Claims 5-8, 13-15 and 22-23 were rejected under 35 U.S.C. 102(e), as anticipated by the patent to Harres et al U.S patent number 6,886,081 (hereinafter referred to as, "Harres"). Furthermore, claims 16-19 were rejected under 35 U.S.C. 103(a) as being unpatentable over Harres in view of Albus et al. U.S. patent number 7,203,823 (hereinafter referred to as "Albus"). Moreover, the specification was objected to due to typographical errors in the specification.

In response to the objection and rejections cited above, the specification has been modified, on the basis of the suggestions set forth in the office action. In addition, claims 2, 7 and 16 have been cancelled without prejudice or disclaimer. Further, claims 1, 3-6, 8-11, 13-15, 17 and 19-23 have been amended.

D.1. Objection

In the office action, claim 4 was objected to due to a typographical error in the claim. The claim has been modified on the basis of the suggestions set forth in the 'Amendments to the Claims' section.

D.2. Rejection under 35 USC § 112

Claim 11 was rejected due to antecedent basis not being adhered to. Claim 11 has been modified to clarify the antecedent basis, as set forth in the 'Amendments to the Claims' section. Further, in the office action, claim 11 was rejected under 35 U.S.C. 112, second paragraph, as failing to particularly point out and distinctly claim the subject matter. The applicant respectfully disagrees with the rejection of claim 11 and would like to point out the following:

Claim 11 recites the sequential execution of the plurality of errands till the itinerary is completely executed. Further, clause (a) of Claim 11 recites the loading of an errand for execution, according to the sequence of errands in an errand function list. The errand function list is a list of pointers to the errands to be called for execution in an itinerary. Loading an errand comprises identifying and calling the function pointed by that errand. The support for this recitation is found on page 10, lines 18-25.

Clause (b) of claim 11 recites running the loaded errand and receiving a return value of true or false on the execution of the function pointed by the errand being executed. A return value of true indicates the successful execution of the function, and a return value of false indicates the incomplete execution and blocking of the function.

The support for this recitation is found on page 14, lines 1-7.

Clause (c) of claim 11 recites loading and executing of the next errand of the itinerary based on the errand function list, if a return value of true is received from the errand being executed. The support for this recitation is found on page 14, lines 8-13 & 19-23.

Clause (d) of claim 11 recites blocking of an itinerary for further execution if a return value of false is received from the errand being executed. The support for this recitation is found on page 14, lines 1-7 & 24-26.

Clause (e) of claim 11 recites resumption of the execution of the blocked itinerary when the itinerary is scheduled again. The execution of the itinerary is resumed from the errand blocked during the previous execution of the blocked itinerary. The support for this recitation is found on page 14, lines 24-29 and page 15, lines 1-3.

Claims 16 and 20-23 were rejected under 35 U.S.C. 112, as failing to invoke 35 U.S.C. 112 6th paragraph in using "means-plus-function" language. In response, claim 16 has been cancelled without prejudice or disclaimer. Further, in order to make claims 20-23 comply with the written description, claims 20-23 have been amended, as set forth in the 'Amendments to the Claims' section.

Claims 22-23 were rejected on the ground that the specification did not provide a basis for the "computer product recitation". In order to correct this a paragraph has been added to the specification which is based on the recitations of claims 22-23 as filed. As the claims form part of the specification this added paragraph does not constitute new

matter.

D.3. Rejection under 35 USC § 102

Claims 1-2 were rejected under 35 U.S.C 102 (e), as anticipated by Ignatius. In response, claim 2 has been cancelled without prejudice or disclaimer. However, the applicant respectfully disagrees with the rejection of claim 1 and would like to point out the following differences between Ignatius and the present invention:

Claim 1 recites a method for programming a multithreaded application to minimize thread switching overheads and memory usage while a multithreaded application is being processed. The method describes the writing of a *plurality of errands that correspond to a thread of the application*, where errands are functions that perform specific tasks that constitute the thread. Subsequently, *one or more itineraries are formed that comprise the plurality of errands* corresponding to the thread.

With regard to the clause (a) of claim 1, Ignatius teaches a method for high-speed transfer of data from one location to another, which may be on the same or different computers within a network, using DataPipe mechanism (column 3, lines 25-38). The DataPipe mechanism divides data into logical tasks that can be performed in parallel, such as compression, encryption and content analysis. The logical tasks are then sequenced in the order in which they act in the data (column 3, lines 47-56).

Ignatius fails to teach or suggest the processing of a thread by using a series of errands. The errands are specific tasks the operating system performs on behalf of the thread. The errands of a thread collectively constitute the entire functionality of the

thread. The support for this recitation is found on page 10, lines 10-14.

With regard to the clause (b) of claim 1, Ignatius also teaches that the Datapipe method divides the data to be transferred into subtasks, and sequences them according to their logical order of processing. The method uses dedicated processes or threads to perform network transfer. All the method tasks share a single large memory segment that is split into small buffers. A single buffer space is shared by the execution threads at different stages of the tasks. The method also uses operating system specific mutual exclusion or signaling primitives, for example, semaphores, to transfer control over the data segment between the modules (column 4, lines 8-24). *Ignatius fails to teach the concept of forming itineraries corresponding to the errands of a thread.* The itinerary instructs the operating system to execute the errands in a logical sequence. The support for this recitation is found on page 10, lines 15-25.

With regard to claim 1, Ignatius teaches a method for transferring data from one location to another. However, claim 1 of the present invention does not concentrate on transferring data from one location to other but rather recites a method for forming itineraries from a plurality of errands, where the plurality of errands are small tasks executed by the operating system, which constitute the itinerarized thread.

Claims 3-4 were objected to as being dependent upon a rejected base claim 1 but were allowable if written in independent form including all the limitations of the base claim. The applicant gratefully acknowledges the allowable matter. In order to make claims 3 and 4 comply with the written description, the claims have been amended, as set forth in the 'Amendments to the Claims' section. However, in view of the above

arguments and when read in conjunction with claim 1, claim 3 teaches a method of forming an itinerary of errands. Forming an itinerary includes storing information regarding the plurality of errands, sequence of execution of errands and data required for execution of the errands. Further, claim 4 read in conjunction with claims 1 and 3, teaches a method for defining errand state field which indicate the state of the errand that is being executed. For example, errand state field for an errand called for the first time for execution may be set to NEW_ERRAND. However, in light of the arguments above, Ignatius fails to teach a method for programming a multithreaded application using errands and itineraries and concentrates on a method for transferring data between two locations.

Claims 5-8, 13-15 and 22-23 were rejected under 35 U.S.C 102 (e), as anticipated by Harres. In response, claim 7 has been cancelled without prejudice or disclaimer. Further, in order to make claims 5-6, 8, 13-15 and 22-23 comply with the written description, the claims have been amended, as set forth in the 'Amendments to the Claims' section. However, the applicant respectfully disagrees with the rejection of claim 5-6, 8, and 22-23 and would like to point out the following differences between Harres and the present invention:

With regard to claim 5, Harres teaches the management of shared resources in multithreaded processing. Further, it suggests a method for determining a set of owners for a lock to a system resource that can be shared by a number of threads. The set of owners is a set of threads that can occupy the resource for execution (column 2, lines 50-67 and column 3, lines 1-2). *However, Harres fails to teach or suggest a method for*

scheduling a plurality of itinerarized threads for execution. The plurality of threads includes *standard threads and itinerarized threads.* The itinerarized threads include the itineraries of the errands associated with the threads. The support for this recitation is found on page 9, lines 13-20. Further, Harres does not teach or suggest a method for writing the plurality of errands for a thread and scheduling the plurality of errands for execution.

With regard to clause (c) of claim 5, Harres teaches that "thread" refers to a simple execution path through application software, which has its own stack in the memory that contains the execution history of the thread. Further, it suggests uninterrupted processing of the threads using mutex locks (column 1, lines 33-54). *Harres fails to teach or suggest the concept of the plurality of threads that includes standard and itinerarized threads. The itinerarized threads may include standard thread constructs and itineraries.* Processing of the itinerarized threads requires processing of the standard thread construct and the itineraries. The standard threads and the standard thread constructs of the itinerarized threads are processed in accordance with the standard execution methodology. The support for this recitation is found on page 9, lines 21-29 and page 10, lines 1-9.

With regard to clause (d) of claim 5, Harres teaches that the operating system is divided into a user space and a kernel space. The user space is used to execute the threads according to the thread schedule, and each thread has a stack of memory associated with pointers to the locks or resources it could own and block, to be used by other threads. The kernel space includes a scheduler for scheduling the threads for

execution (column 4, lines 60-67 and column 5, lines 1-13). *In contrast, claim 5 of the present invention recites a method for executing the itineraries in the itinerary mode.*

The support for this recitation is found on page 10, lines 1-9. The itineraries are instructions for the operating system on the sequence of processing errands included in the itineraries. The itinerary mode execution of the itinerarized thread carried out on the kernel stack. The support for this recitation is found on page 11, lines 15-24 and page 15, lines 9-13.

With regard to clause (e) of claim 5, Harres teaches exiting the scheduling mode when scheduling is completed, and continuing the normal thread execution process. In contrast, claim 5 of the present invention recites exiting the itinerary mode execution by *an itinerarized thread after the corresponding itinerary has been executed completely.* Thereafter, the itinerarized thread is executed in the normal mode. The support for this recitation is found on page 14, lines 19-23.

With regard to claim 6, Harres suggests that to execute a critical section of code or access shared data, the threads attempt to acquire a lock or resource for processing. While the thread executes a section of a code, or accesses data after acquiring the lock, other threads are prevented from accessing the lock. The kernel maintains a list of threads that are waiting for the lock for processing (column 1, lines 55-67 and column 2, lines 1-5). However, claim 6 of the present invention recites that when an itinerarized thread, executing in the normal mode, makes a request to run an itinerary, the normal mode execution of the itinerarized thread is preempted. Subsequently, the itinerarized thread is executed in the itinerary mode. The support for this recitation is found on page

12, lines 7-24.

The dependency of claim 8 has been changed to make it dependent on independent claim 5. The dependent claim 8 has also been amended to recite the execution of the plurality of threads in the normal mode. With regard to claim 8, Harres teaches loading and executing a standard thread according to scheduling by the kernel. Further, Harres teaches a method of preempting the thread being executed on the lock, and suggests the need for storing ownership information pertaining to the multiple owner lock in column 1, lines 55-67, column 2, lines 1-5 and lines 31-43 and column 3, lines 3-21). However, Harres does not teach *loading, execution and preemption of an itineraryarized thread in the normal mode execution for itinerary mode execution*. The support for this recitation is found on page 10, lines 1-9 and page 12, lines 7-24.

Claims 9-12 were objected to as being dependent upon a rejected base claim 5 but were allowable if written in independent form including all the limitations of the base claim. The applicant gratefully acknowledges the allowable subject matter. In order to make claims 9-11 comply with the written description, the claims have been amended, as set forth in the 'Amendments to the Claims' section. However, in view of the above arguments and when read in conjunction with claim 5, claim 9 teaches a method of executing an itinerary of an itineraryarized thread. Executing an itinerary includes forming an itinerary of errands by sequencing the errands and executing the errands according to the sequence. Further, claim 10 read in conjunction with claims 5 and 9, teaches that a conditional errand can change the sequence of execution of the errands in the itinerary. Furthermore, claim 11 read in conjunction with claims 5 and 9, teaches a

method of sequentially executing the errands of the itinerary and blocking the itinerary when any of the errands fails to execute completely. Moreover, claim 12 read in conjunction with claims 5, 9 and 11, teaches defining and modifying the errand state field of the errand being executing depending upon if the errand is executed completely or blocked. However, in light of the arguments above, Harres fails to teach a method for executing a multithreaded application using itinerarized threads comprising itineraries and errands and concentrates only on executing a standard thread.

With regard to clause (b) of claim 13, Harres teaches a system for determining the ownership of a lock, for the execution of the plurality of threads. Further, Harres suggests a system for effective resource management, for parallel processing of the threads. However, Harres does not teach or suggest *storing information related to the plurality of threads*. This information includes the thread stack and context information relating to the standard threads and standard thread constructs of the itinerarized threads, and the kernel stack pertaining to the itineraries of the itinerarized threads. The support for this recitation is found on page 9, lines 21-29 and page 10, lines 1-9.

With regard to clause (c) of claim 13, Harres suggests that the system may fail to operate efficiently due to the execution of one or more processes that block the resource for access by other processes (column 1, line 17-32). In contrast, claim 13 recites a processor for processing the plurality of threads by accessing the related information stored in the thread stack and the kernel stack of the memory. The support for this recitation is found on page 9, lines 21-29 and page 10, lines 1-9.

With regard to clause (d) of claim 13, Harres suggests the importance of an

operating system, which includes device drivers, memory management routines, schedulers and system calls to correct bugs in a crashed computer system (column 2, lines 3-22). Claim 13 of the present invention emphasizes the importance of the operating system for scheduling and managing the execution of the plurality of threads. The operating system executes the standard threads and the standard thread construct of the itinerarized threads according to the standard thread execution methodology. Further, the operating system *executes the itineraries of the itinerarized threads in the itinerary mode*. The support for this recitation is found on page 9, lines 21-29 and page 10, lines 1-9.

With regard to claim 14, Harres teaches that each thread has a thread stack associated with it in the memory that contains the thread execution history. However, claim 14 of the present invention recites that a *separate kernel stack* is associated with each processor of the multiprocessor system. The kernel stack is used to *execute the itinerary of an itinerarized thread* in the itinerary mode. The support for this recitation is found on page 10, lines 1-9.

With regard to clause (a) of claim 15, Harres suggests that the operating system includes a scheduler for scheduling the processing of the plurality of threads. However, claim 15 of the present invention recites an operating system that includes a scheduler for scheduling the *processing of the standard and the itinerarized threads*. Further, the present invention as defined in claim 15 provides a standard thread running service to execute the standard threads and standard thread constructs of the itinerarized threads. Furthermore, the present invention provides as defined in claim 15 *an itinerary running*

service for executing the itinerarized threads in the itinerary mode. The support for this recitation is found on page 9, lines 13-20.

With regard to clause (b) of claim 15, Harres suggests a standard thread-running service for executing the standard threads (column 1, lines 33-54). Claim 15 recites the *execution of the standard constructs of the itinerarized threads* by the standard thread running service. The support for this recitation is found on page 9, lines 13-20 and page 10, lines 1-9.

With regard to clause (c) of claim 15, Harres suggests a scheduler for scheduling the execution of the plurality of threads (column 1, lines 33-54). Claim 15 recites that the *itinerary running service schedules the itineraries* of the itinerarized threaded for execution in the itinerary mode. The support for this recitation is found on page 10, lines 1-9.

Claims 22-23 are the computer program product variant of claims 5-6 and 8-9. The same arguments hold true for them as for their respective method claims.

Further, claims 20-21 were rejected under 35 U.S.C 102 (e), as anticipated by Albu. In order to make claims 20-21 comply with the written description the claims have been amended, as set forth in the 'Amendments to the Claims' section. However, the applicant respectfully disagrees and would like to point out the following differences between Albu and the present invention:

Claim 20 recites an itinerary running service for executing and itinerarized thread in the itinerary mode. The itinerary running service includes preemptive and non-

preemptive services to *preempt the itineraryized thread for executing the thread in the itinerary mode*. Further, the itinerary running service includes the *itinerary building service* to build itineraries from the errands.

With regard to clause (a) and (b) claim 20, Albuze teaches the operation of preemptive and non-preemptive threads when they are interrupted by an event that signals the execution of another thread. In the preemptive threads, the execution is interrupted and the resource is made available to the preempting thread, whereas in the non-preemptive threads, execution is continued till the end of processing (column 5, lines 30-62). However, Albuze does not teach the execution of *preemptive and non-preemptive errands*. Errands are functions or subtasks executed by the operating system to process the corresponding thread. The present invention in claim 20 describes an itinerary-running services that includes a preemptive service, a non-preemptive service and an itinerary building service to execute the itinerary of an itineraryized thread in the itinerary mode. The various preemptive and non-preemptive services include standard errands provided by the system. The preemptive service *may preempt the itinerary of the itineraryized thread* being executed when a preemption request is received from a preempting thread. The non-preemptive service does not preempt execution of the itinerary of the itineraryized thread when a preemption request is received from a preempting thread. The support for this recitation is found on page 10, lines 26-30, page 11, lines 1-5 and lines 15-24.

With regard to clause (c) of claim 20, Albuze teaches the execution of start-over threads that are non-preemptive threads, which occur frequently for a short duration.

When the start-over thread interrupts a non-preemptive thread, no preemption takes place. The non-preemptive thread calls rescheduling after its complete execution. The scheduler identifies that the start-over thread is the highest priority thread, and consequently, executes the start-over thread. The itinerary building service of the present invention schedules pre-programmed standard errands and application specific errands, for execution by the operating system. The pre-programmed standard errands provided by the system constitute the various preemptive and non-preemptive services. The application specification errands are small tasks that constitute a thread to be executed. Further, the itinerary building service executes the errands sequentially according to the schedule. Furthermore, the errand building service blocks the execution of the itinerary when one of the scheduled errands is blocked for execution. The support for this recitation is found on page 18, lines 1-10.

With regard to claim 21, Albuz teaches the operation of preemptive and non-preemptive threads when interrupted by an event that signals the execution of another thread. In the preemptive threads, the execution is interrupted and the resource is made available to the preempting thread, whereas in the non-preemptive threads, the execution is continued till the end of processing (column 5, lines 30-62). However, Albuz fails to teach the resumption of the execution of a previously blocked itinerary from the errand that blocked the execution of the itinerary when the itinerary is scheduled back on the itinerary running service. The support for this recitation is found on page 14, lines 24-29 and page 14, lines 1-3.

D.4. Rejection under 35 USC 103

In response to the rejections over claims 16-19 under 35 U.S.C 103 (a), which form the basis of obviousness, claim 16 has been cancelled without prejudice or disclaimer. Further, in order to make claims 17-19 comply with the written description, they have been amended, as set forth in the 'Amendments to the Claims' section. However, the applicant respectfully disagrees with the objection over claims 17-19, based on the combination of Harres and Albuza, and would like to point out the following differences between the combination of Harres and Albuza and the present invention:

The dependency of claim 17 has been amended to make it dependent on independent claim 15. With regard to claim 17, Albuza teaches that the execution of start-over threads, which are non-preemptive threads, occurs frequently for a short duration. When the start-over thread interrupts a non-preemptive thread, no preemption takes place. The non-preemptive thread calls rescheduling after its complete execution. The scheduler identifies that the start-over thread is the highest priority thread, and consequently, executes it (column 7, lines 66-67 and column 8, lines 1-24). However, Albuza fails to teach the maintenance of separate ready queues for the execution of the standard threads and the itinerarized threads. The present invention is directed to scheduling of an itinerarized thread in the ready queue after it completes execution in the itinerary mode. The support for this recitation is found on page 9, lines 13-24 and page 12, lines 15-24.

Further, the applicant respectfully disagrees with the objection on claims 18-19, based on the combination of Harres and Albuza, in view of the arguments given above,

which make clear the fact that the method steps of Albu and the present invention are different. Therefore, with regard to claim 18 and 19, Harres and Albu, combined, do not teach or suggest *the execution of the itinerarized thread, partially on the standard thread running service and partially on the itinerary running service*. Furthermore, Harres and Albu, combined, do not teach or suggest the itinerary running service, as depicted in the present invention (column 5, lines 30-62). Moreover, Albu does not teach the execution of preemptive and non-preemptive errands. The present invention is directed to itinerary running services including a preemptive service, a non-preemptive service and an itinerary building service for executing the itinerary of an itinerarized thread in the itinerary mode. The various preemptive and non-preemptive services include standard errands provided by the system. The preemptive service may preempt the itinerary of the itinerarized thread being executed when a preemption request is received from a preempting thread. The non-preemptive service does not preempt the execution of the itinerary of the itinerarized thread when a preemption request is received from a preempting thread. The support for this recitation is found on page 10, lines 26-30, page 11, lines 1-5 and lines 15-24.

CONCLUSION

In view of the present amendment, it is submitted that the claims are patentably distinct over the cited art, and that the rejections to the claims have been overcome, and notice to that effect is earnestly solicited. Accordingly, reconsideration of the present application is requested. If the Examiner has any questions regarding this matter, the

Examiner is requested to telephone the applicant's attorney at the numbers listed below, prior to issuing a further office action.

Dated: November 26, 2007

Respectfully submitted,

By 
William L. Botjer
Reg. No. 27,990
PO Box 478
Center Moriches, NY 11934
(212) 737-5728 (Tue-Thurs)
(631) 874-4826 (Mon & Fri)
(631) 834-0611 (cell if others busy)